PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2002-183750
(43)Date of publication of application : 28.06.2002

(51)Int.Cl. G06T 15/00
G06F 9/38
G06T 1/20

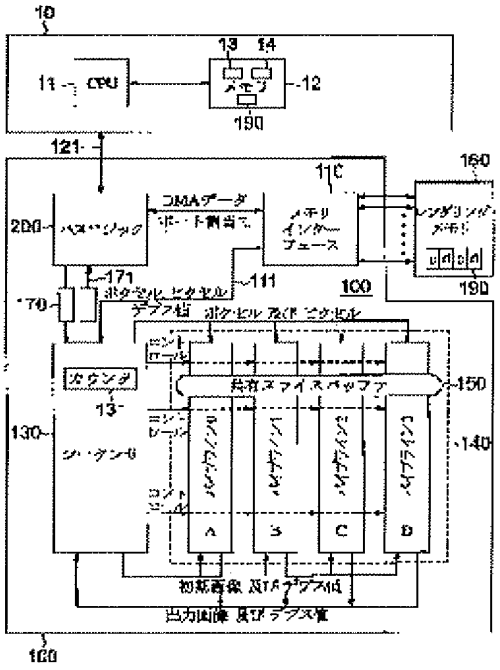
(21)Application number : 2001-305961
(22)Date of filing : 02.10.2001

(71)Applicant : TERARIKON INC
(72)Inventor : YEAGER MARK O
CORRELL KENNETH W
KERZNER STEVEN
LAUER HUGH C
SEILER LARRY D

(30)Priority
Priority number : 2000 679248 Priority date : 04.10.2000 Priority country : US

(54) RENDERING SYSTEM AND RENDERING METHOD

(57)Abstract:
PROBLEM TO BE SOLVED: To provide a rendering system capable of temporally superimposing the rendering operation over data transfer operation for a long period, and synchronizing individual operations with each other.
SOLUTION: This rendering system for executing rendering of graphic data has a host processor generating a data transfer command and a rendering command, and a memory storing the data transfer command and the rendering command in a command queue. A command parser of the system simultaneously parses and process the data transfer command and the rendering command of the command queue, and a synchronization mechanism synchronizes the processing with the parsing of the data transfer command and the rendering command, simultaneously executed.



(19)日本国特許庁（ＪＰ）　(12) **公開特許公報**（Ａ）　(11)特許出願公開番号
特開2002－183750
（P2002－183750A）
(43)公開日　平成14年６月28日(2002.6.28)

(51)Int.Cl.⁷	識別記号	ＦＩ		テーマコード（参考）
Ｇ０６Ｔ　15/00	２００	Ｇ０６Ｔ　15/00	２００	５Ｂ０１３
Ｇ０６Ｆ　9/38	３７０	Ｇ０６Ｆ　9/38	３７０Ａ	５Ｂ０５７
Ｇ０６Ｔ　1/20		Ｇ０６Ｔ　1/20	Ｃ	５Ｂ０８０

審査請求　未請求　請求項の数16　ＯＬ　（全 14 頁）

(21)出願番号　特願2001－305961(P2001－305961)

(22)出願日　平成13年10月２日(2001.10.2)

(31)優先権主張番号　０９／６７９２４８

(32)優先日　平成12年10月４日(2000.10.4)

(33)優先権主張国　米国（ＵＳ）

(71)出願人　599059933
テラリコン・インコーポレイテッド
アメリカ合衆国　カリフォルニア州サンマテオ市キャンパスドライブ2955スイート325

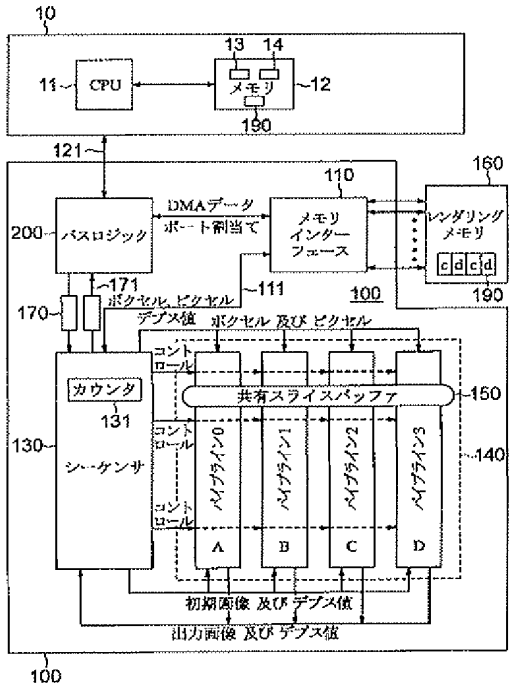
(72)発明者　マーク・オー・イーガー
アメリカ合衆国、マサチューセッツ州、ウエスト・グラトン、オールド・オーチャード・ストリート　21

(74)代理人　100057874
弁理士　曾我　道照　（外６名）

最終頁に続く

(54)【発明の名称】　レンダリングシステム及びレンダリング方法

(57)【要約】
【課題】　長時間にわたるレンダリング動作およびデータ転送動作を時間的に重複させることができ、かつ、個々の動作を互いに同期させることができるレンダリングシステムを得る。
【解決手段】　グラフィックデータをレンダリングするためのレンダリングシステムは、データ転送コマンドとレンダリングコマンドとを生成するホストプロセッサと、コマンドキューにデータ転送コマンドとレンダリングコマンドとを格納するメモリとを備える。そのシステムのコマンドパーサは、そのコマンドキューのデータ転送コマンドとレンダリングコマンドとを同時に構文解析し、かつ処理し、同期機構を用いて、同時に行われるデータ転送コマンドおよびレンダリングコマンドの構文解析および処理を同期させる。



【特許請求の範囲】

【請求項１】　レンダリングシステムであって、
複数のコマンドキュー内にデータ転送コマンドとレンダリングコマンドとを格納するメモリと、
前記複数のコマンドキューそれぞれのためのコマンドキューロジックであって、各コマンドキューロジックは、対応するコマンドキューの前記データ転送コマンドおよび前記レンダリングコマンドを同時にフェッチし、構文解析し、処理する、該コマンドキューロジックと、
グラフィックデータをレンダリングしている間に、同時に行われる前記データ転送コマンドおよび前記レンダリングコマンドの前記フェッチ、前記構文解析および前記処理を同期させるための手段とを備えるレンダリングシステム。

【請求項２】　前記複数のコマンドキューに格納された同期コマンドが前記複数のコマンドキューを同期させる請求項１に記載のレンダリングシステム。

【請求項３】　前記同期コマンドは、前記レンダリングシステムに接続されるホストプロセッサによって生成される請求項２に記載のレンダリングシステム。

【請求項４】　前記同期コマンドは、前記レンダリングシステムのレンダリングパイプラインによって生成される請求項２に記載のレンダリングシステム。

【請求項５】　前記コマンドキューのうちの少なくとも１つは、ホストメモリに格納される請求項１に記載のレンダリングシステム。

【請求項６】　前記コマンドキューのうちの少なくとも１つは、レンダリングメモリに格納される請求項１に記載のレンダリングシステム。

【請求項７】　前記複数のキューは、前記データ転送コマンドと前記レンダリングコマンドとを格納するために、複数の要素を有する循環バッファとして構成される請求項１に記載のレンダリングシステム。

【請求項８】　前記各コマンドキューロジックはさらに、
前記対応するコマンドキューから前記データ転送コマンドと前記レンダリングコマンドとをフェッチするためのＤＭＡ状態機械と、
前記データ転送コマンドと前記レンダリングコマンドとを構文解析し、処理し、さらに同期させるためのパース状態機械とを備える請求項１に記載のレンダリングシステム。

【請求項９】　前記データ転送コマンドはＤＭＡインターフェースに発行され、前記レンダリングコマンドは複数の並列パイプラインに発行される請求項１に記載のレンダリングシステム。

【請求項１０】　前記パース状態機械は、リセット状態と、停止状態と、実行状態と、コマンド待ち状態と、同期待ち状態とを有する請求項８に記載のレンダリングシステム。

【請求項１１】　イベントを格納するためのスクラッチレジスタをさらに備え、前記同期コマンドは、特定のイベントの発生のために前記スクラッチレジスタを検査する請求項２に記載のレンダリングシステム。

【請求項１２】　前記データ転送コマンドおよび前記レンダリングコマンドの構文解析および処理は、前記特定のイベントの発生を待つ間、停止される請求項１１に記載のレンダリングシステム。

【請求項１３】　前記各コマンドキューは前記データ転送コマンドと前記レンダリングコマンドとを格納するための複数のエントリを含み、
第１のポインタを、フェッチされることになる次のコマンドを格納する第１のエントリに格納するための、前記各コマンドキューのためのプロデューサレジスタと、
第２のポインタを、次のコマンドを格納するための第２のエントリに格納するための、前記各コマンドキューのためのコンシューマレジスタとをさらに備える請求項１に記載のレンダリングシステム。

【請求項１４】　レンダリングシステムであって、
複数のコマンドキューにデータ転送コマンドと、レンダリングコマンドと、同期コマンドとを格納するように構成されるメモリと、
イベントを格納するように構成される複数のレジスタと、
前記複数のコマンドキューそれぞれのためのコマンドキューロジックであって、各コマンドキューロジックは、対応するコマンドキューのデータ転送コマンドとレンダリングコマンドとを同時にフェッチし、構文解析し、処理し、また前記コマンドキューロジックは、データをレンダリングする間に、同期コマンドを用いて前記複数のレジスタに格納される特定のイベントのための検査を行うことにより、同時に行われる前記データ転送コマンドと前記レンダリングコマンドとのフェッチ、構文解析および処理を同期させる、該コマンドキューロジックとを備えるレンダリングシステム。

【請求項１５】　レンダリング方法であって、
複数のコマンドキューにデータ転送コマンドとレンダリングコマンドとを格納するステップと、
対応するコマンドキューのデータ転送コマンドとレンダリングコマンドとを同時にフェッチし、構文解析し、処理するステップと、
グラフィックデータをレンダリングする間に、同時に行われる前記データ転送コマンドと前記レンダリングコマンドとの前記フェッチ、前記構文解析および前記処理を同期させるステップとを含む方法。

【請求項１６】　レジスタにイベントを格納するステップと、
同時に行われる前記データ転送コマンドと前記レンダリングコマンドとの前記フェッチ、前記構文解析および前記処理を同期させるために、同期コマンドを用いて、特

定のイベントの発生のために前記レジスタを検査するステップとをさらに含む請求項１５に記載の方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明はコンピュータグラフィックスの分野に関し、特に、並列パイプライン式のレンダリングシステムにおいてグラフィックデータをレンダリングすることに関する。

【０００２】

【従来の技術】コンピュータグラフィックスアプリケーションにおいて、３次元データを視覚化する必要があるとき、多くの場合にボリュームレンダリングが用いられる。そのボリュームデータには、物理的または医学的対象物、あるいは大気、地理あるいは他の科学的モデルの走査データを用いることができ、その場合には、データの視覚化によって、データによって表される根底をなす現実世界の構造を理解しやすくなる。

【０００３】ボリュームレンダリングを用いると、物理的な物体およびモデルの内部構造ならびに外部表面の特徴が視覚化される。ボクセルは通常、ボリュームレンダリングにおいて用いられる基本的なデータ項目である。ボクセルは、その物体あるいはモデルの特定の３次元部分に関連する。各ボクセルの座標（ｘ，ｙ，ｚ）は、ボクセルを、表現される物体あるいはモデル内の位置にマッピングする。

【０００４】ボクセルは、その物体あるいはモデルの特定の位置に関連する１つあるいは複数の値を表す。従来技術によるボリュームレンダリングシステムでは、ボクセルによって表される値には、密度、組織タイプ、弾性あるいは速度のような多数の異なるパラメータのうちの特定の１つのパラメータを用いることができる。レンダリング中に、ボクセルは、その強度値に応じた色および不透明度（ＲＢＧα）値に変換される。その値は、視認するために２次元の画像平面上に投影することができる。

【０００５】レンダリング中に頻繁に用いられる１つの技術はレイキャスティングである。その場合、１組の仮想的なレイ（視線）が、ボクセルのアレイを通して投影される。レイはいくつかの視点あるいは画像平面から生じる。ボクセルの強度値はレイに沿った点でサンプリングされ、サンプリングされる値を画素値に変換するために種々の技術が知られている。

【０００６】Burgess等による１９９９年５月２０日出願の米国特許出願第０９／３１５，７４２号、タイトル「Volume rendering integrated circuit」が、参照して本明細書に援用される。そこでは、簡単な従来技術のボリュームレンダリングシステムが記載される。そのレンダリングシステムは、相互接続バスによってボリュームグラフィックスボード（ＶＧＢ）に接続されるホストプロセッサを備える。そのホストプロセッサには、バスを含む任意の種類のパーソナルコンピュータあるいはワークステーションを用いることができる。そのホストは、ホストメモリあるいはメインメモリを備える。ホストメモリには、メインメモリ、キャッシュメモリおよびディスクメモリのような、プロセッサが利用可能な内部あるいは外部の記憶装置の任意の組み合わせを用いることができる。

【０００７】ＶＧＢはボリュームレンダリングチップ（ＶＲＣ）に接続されるボクセルメモリとピクセルメモリとを備える。ＶＲＣは、リアルタイムでインタラクティブなボリュームレンダリング動作に必要な全てのロジックを備える。ＶＲＣは、４つの相互に接続されるレンダリングパイプラインを含む。基本的には、ＶＧＢはホストのためのレンダリングエンジンあるいは「グラフィックスアクセラレータ」を提供する。

【０００８】動作中に、ホスト内で実行されるアプリケーションソフトウエアが、レンダリングのために、ボリュームデータをＶＧＢに転送する。詳細には、ボクセルデータは、ホストメモリからバスを介してボクセルメモリに転送される。またアプリケーションは、ボクセルメモリ内の分類テーブルのような別のデータも格納する。またアプリケーションは、パイプラインによってアクセス可能なレンダリングレジスタをロードする。これらのレジスタは、そのレンダリングが如何に実行されることになるかを指定する。全てのデータがロードされた後に、アプリケーションは、レンダリング動作を開始するためにコマンドを生成する。パイプラインはそのレンダリングコマンドを実行する。レンダリング動作が完了すると、出力装置上で表示するために、出力画像がピクセルメモリからホストあるいは３Ｄグラフィックスカードに移動する。Burgessのシステムでは、主な動作は逐次的に行われる。すなわち、そのシステムは、ホストからＶＧＢにボクセルデータを書き込み、ＶＲＣにおいてレンダリングを実行し、その後、ＶＧＢからホストにピクセルデータを書き込む。

【０００９】しかしながら、さらに複雑なレンダリング動作の場合、メモリ転送とレンダリング動作とを重複して行うか、あるいは以前のレンダリング動作が依然として進行している間に、１つのレンダリング動作を開始することが望ましい場合がある。さらに、効率上の理由により、ホストにおいて実行されているアプリケーションあるいは他のソフトウエアによって割り込まれたり、介入されたりすることなく、あるレンダリング動作から次のレンダリング動作まで進行することが望ましい場合がある。同様に、ソフトウエアによる割込みあるいは介入が生じることなく、レンダリング動作の完了からデータ転送の開始まで進行することが望ましい場合がある。

【００１０】たとえば、パイプラインが現在のボリュームをレンダリングしている間に、以前のボリュームのレンダリングから生成された画像をホストに転送して戻

し、レンダリングされることになる次のボリュームをロードすることができるであろう。この状況では、次のボリュームのためのレンダリング動作は、そのボリュームが完全にロードされるまで開始することを禁止され、画像のデータ転送は、その画像を生成するレンダリング動作が完了するまで開始してはならない。この状況は、データの移動が、埋め込まれた多角形構造を含むときに、急激に、さらに一層複雑になる。なぜなら、これが、レンダリングを開始する前にロードされなければならない種々のデータ源の数を増加させるためである。

【００１１】

【発明が解決しようとする課題】それゆえ、本発明の目的は、長時間にわたるレンダリング動作およびデータ転送などの動作を時間的に重複させることができ、ホストソフトウエアが介入することなく、個々の動作を互いに同期させることができ、さらにホストソフトウエアがレンダリング動作の状態を判定し、進行中の動作と同期できるように、レンダリングシステムを制御するための効率的な手段を提供することである。

【００１２】

【課題を解決するための手段】本発明は、データ転送コマンドおよびレンダリングコマンドを生成するホストプロセッサを備える、グラフィックスデータをレンダリングするためのレンダリングシステムを提供する。またそのシステムは、データ転送コマンドおよびレンダリングコマンドを複数のコマンドキューに格納するメモリと、複数のコマンドキューのデータ転送コマンドおよびレンダリングコマンドを同時に構文解析し、かつ処理するコマンドパーサとを備える。同期機構を用いて、データ転送コマンドおよびレンダリングコマンドを同時に構文解析し、かつ処理する動作を同期させる。

【００１３】

【発明の実施の形態】パイプライン機構

図１は本発明によるボリュームレンダリングシステム全体の機構を示す。そのシステムは、バス１２１によってレンダリングサブシステム１００に接続されるホストコンピュータ１０を備える。ホストは、ＣＰＵ１１と、ホストメモリ１２とを備える。ホストメモリ１２は、グラフィカルアプリケーションソフトウエア１３と、オペレーティングシステムソフトウエアと通信するグラフィックドライバ１４とを格納することができる。そのソフトウエアはＣＰＵ１１において実行される。またホストメモリは、ボリュームデータセット、画像、および種々のテーブルのようなレンダリングデータも格納することができる。

【００１４】レンダリングエンジン１００の主なモジュールは、メモリインターフェース１１０、バスロジック２００、シーケンサ１３０、および４つの並列パイプライン１４０である。４つのパイプラインに及ぶ一対のスライスバッファ１５０を除いて、パイプライン（Ａ、Ｂ、ＣおよびＤ）は互いに独立して動作する。好ましい実施形態では、全ての主なモジュール１１０、２００、１３０、１４０および１５０は、１つのＶＬＳＩチップ上のハードウエア回路で実装される。これは、本発明を用いることができるレンダリングエンジンのタイプの一例にすぎないことに留意されたい。レンダリングエンジンには、パイプライン化された、またはパイプライン化されていない多角形レンダリングエンジン、あるいは任意の他のタイプのハードウエアで実装されたレンダリングエンジンを用いることができることにも留意されたい。

【００１５】メモリインターフェース

メモリインターフェース１１０は、レンダリングメモリ１６０を制御する。好ましい実施形態では、レンダリングメモリ１６０は、８つの倍速データレート（ＤＤＲ（double data rate））同期ＤＲＡＭモジュールを備える。そのレンダリングメモリは、ボリューム（ボクセル）、多角形、入力および出力画像（ピクセル）、デプス値（depth values）および参照用テーブルのような、パイプライン１４０内のレンダリンググラフィックス対象物のために直接必要とされる全てのレンダリングデータ１１１のための一体化された記憶装置を提供する。以下に記載されるように、レンダリングメモリ１６０は、本発明によるコマンドキューバッファ１９０を格納することもできる。別法では、ホストＣＰＵメモリ１２がコマンドキューバッファ１９０を格納することができる。すなわち、任意の個々のコマンドキューバッファは、レンダリングメモリ１６０か、あるいはホストメモリ１２かのいずれかに格納することができる。

【００１６】メモリインターフェース１１０は、レンダリングメモリ１６０への全てのメモリアクセスを実施し、バスロジック２００およびシーケンサ１３０の要求を仲裁し、データを、サブシステム１００およびレンダリングメモリ１６０に分散させる。この利点として、広帯域幅メモリアクセス（読出しおよび書込み）が、レンダリング動作と重複して行われる。

【００１７】好ましい実施形態では、ＶＬＳＩ１００およびレンダリングメモリ１６０は、ＰＣＩバス１２１にプラグインすることができる１枚のボードとして実装される。

【００１８】シーケンサ

シーケンサ１３０はレンダリングエンジンを制御する。それは、どのデータがメモリからフェッチされるかを判定し、そのデータを４つのパイプライン１４０にディスパッチし、補間の重みのようなコントロール情報を正確な時間に個々のパイプラインに送信し、レンダリング動作からの出力画像を受信する。シーケンサそのものは、多数のレジスタによって制御される１組の有限状態機械である。これらは典型的には、特定のコマンドキューのロードレジスタコマンドに応答してバスロジック２００

によって書き込まれるが、バスロジックへのＰＣＩアクセスを介して、ホストシステム上のソフトウエアによって直接書き込まれる場合もある。いずれの場合でも、バスロジック２００がＦＩＦＯ１７０を介してレジスタ値をシーケンサ１３０に書き込む。これにより、バスロジック２００の動作は、シーケンサ１３０の動作と切り離すことができるようになり、バスロジックおよびシーケンサが異なるサイクルタイムで動作できるようになる。

【００１９】内部では、シーケンサ１３０は、ある時点でサンプル空間を介して１つのセクションを進め、サンプル座標を順序変更されたボクセル座標に変換し、さらに４つのパイプラインのステージによって必要とされるコントロール情報を生成するために必要とされるカウンタ１３１を保持する。またシーケンサは、バスロジックに、レンダリング動作のステータス１７１も提供する。その後、バスロジックはレンダリングステータスをホスト１０に提供する。

【００２０】バスロジック

図２は、バスロジック２００の内部構造を示す。バスロジック２００は、コマンドキューロジック３００を含み、バス１２１を介してホスト１０へのインターフェースを、またＦＩＦＯ１７０を介してメモリインターフェース１１０およびシーケンサ１３０へのインターフェースを提供する。ホストがパーソナルコンピュータ（ＰＣ）あるいはワークステーションである場合には、バス１２１には、ＰＣＩ仕様バージョン２．２に準拠する６４ビット、６６ＭＨｚＰＣＩバス１２１を用いることができる。

【００２１】このために、バスロジックは、メモリインターフェース１１０を介してレンダリングメモリ１６０にデータ（ＤｍａＩｎ）を転送し、またレンダリングメモリ１６０からデータ（ＤｍａＯｕｔ）を転送するためのダイレクトメモリアクセス（ＤＭＡ）動作を制御するＤＭＡインターフェース（ＤｍａＩｆ）２１０を含む。ＤＭＡ動作は、バーストモードデータ転送である。このために、バスロジックはＰＣＩバスマスタとして動作する。

【００２２】ターゲットインターフェース（ＴａｒｇｅｔＩｆ）２２０は、レンダリングレジスタ２３０の読出しおよび書込みを制御する。これらのアクセスは、個々のレジスタおよびメモリ内の個々の場所に対する直接的な読出しおよび／または書込みであり、ホスト１０によって、あるいはＰＣＩバス上のいくつかの他の装置によって開始される。

【００２３】またバスロジックは、メモリコントロールおよび仲裁ロジック２４０も備え、そのロジックは、ＤＭＡ要求、およびターゲットレジスタ読出し／書込み要求を仲裁する。またロジック２４０は、メモリアクセス動作を、バスロジック２００とメモリインターフェース１１０との間のコマンドに変換する。またバスロジックは、レンダリング動作を制御するために、シーケンサ１３０に直接にレジスタ値を送信し、かつシーケンサ１３０から戻されるステータス１７１を受信する。

【００２４】最後に、バスロジックは、コマンドキューロジック３００の３つの同一のコピー（番号２０１、２０２および２０３）を含む。好ましい実施形態は、３つのコマンドキューに対応する。典型的には、以下に記載されるように、ロジック２０１～２０３は、それぞれＤＭＡｉｎコマンド、レンダリングコマンドおよびＤＭＡｏｕｔコマンドを専用に処理する。好ましい実施形態では、コマンドキューリングバッファ１９０が、レンダリングメモリ１６０あるいはホストメモリ１２に格納される。別法では、そのバッファはコマンドキューロジック３００の一部にすることができる。

【００２５】コマンドキューロジック

図２に示されるように、そのレンダリングシステムは、３つのコマンドキュー３００、すなわちＤｍａＩｎ２０１、レンダー（Ｒｅｎｄｅｒ）２０２およびＤｍａＯｕｔ２０３に対応する。典型的な使用法では、ＤｍａＩｎコマンドキュー２０１は、ホストメモリ１２からレンダリングメモリ１６０へのデータの転送を制御する。レンダリングコマンドキュー２０２は、ユーザが入力したパラメータにしたがってレンダリングデータ１１１をレンダリングする。それは、レンダリングメモリ１６０内の場所間のグラフィックスデータをコピーすることを含む。ＤｍａＯｕｔコマンドキュー２０３は、レンダリングメモリ１６０からホストメモリ１２へのデータの転送を制御する。このデータには、画像あるいは部分的に処理されたレンダリングデータを用いることができる。各コマンドキューバッファ１９０は、ホストメモリ１８０内、あるいはレンダリングメモリ１６０内に存在することができる。

【００２６】図３は、各コマンドキュー２０１～２０３を実装するロジック３００を示す。各コマンドキューに対して、コマンドキューロジック３００の１つのコピーが存在する。ロジック３００は、コマンドキュー状態レジスタ３１０、ＤＭＡ状態機械３２０、パース状態機械３３０、レジスタアレイ３４０、ポインタロジック３５０、およびステータスロジック３６０を備える。また、各コマンドキューロジックは、共有されるステータスレジスタ６００にステータス情報を報告する。そのロジックの３つの全てのコピーに対してステータスレジスタは１つしか存在しないことに留意されたい。ステータスレジスタはレンダリングレジスタ２３０の１つである。

【００２７】状態レジスタ３１０は、レンダリングメモリ１６０あるいはホストメモリ１２内の関連するコマンドキューバッファ１９０の場所、およびバッファ内の現在の位置を含む、関連するコマンドキューの状態を規定する。具体的には、そのレジスタは、サイズ（Ｓｉｚｅ）レジスタ３１１、サブコンシューマ（ｓｕｂＣｏｎ

ｓｕｍｅｒ）レジスタ３１２、サブベース（ＳｕｂＢａｓｅ）レジスタ３１３、ベース（Ｂａｓｅ）レジスタ３１４、コンシューマ（Ｃｏｎｓｕｍｅｒ）レジスタ３１５、プロデューサ（Ｐｒｏｄｕｃｅｒ）レジスタ３１６およびスクラッチ（Ｓｃｒａｔｃｈ）レジスタ３１７である。

【００２８】コマンドキューバッファ１９０に格納されるデータ値は、データ転送、レンダリング動作を制御し、転送コマンドとレンダリングコマンドとの同期を可能にする。それゆえ、バッファは、コマンド（演算子）およびデータ（オペランド）を要素ｃおよびｄとして格納する。アプリケーションソフトウエアあるいは他の手段により、要素ｃおよびｄは、コマンドキューバッファ１９０内の空きエントリに順次書き込まれる。異なるコマンドｃは、そのコマンドに関連する異なる量のデータｄを有する場合があり、データを全く持たない場合もある。

【００２９】ＤＭＡ状態機械３１０は、パース状態機械３３０が要素を読み出す、あるいは「消費する」間に、その要素を、コマンドキューバッファ１９０からローカル記憶装置、すなわちレジスタアレイ３４０に読み出す。コマンドキューバッファ１９０が空であるか、あるいは処理されているコマンドｃが、まだ発生していないイベントに同期することを必要とする場合には、関連するプロセスが中止され、さらに多くの要素がそのキューに書き込まれるか、あるいはそのイベントが発生するまで待機する。

【００３０】好ましい実施形態では、各コマンドキューバッファ１９０は、レンダリングメモリ１６０内、あるいは、ホストメモリ１２のようなＰＣＩアドレス空間内のいくつかの外部メモリ内のいずれかに線形アレイとして構成される。コマンドｃおよびデータｄ要素は、線形アレイを通して、１ステップずつインクリメントしながら書込みおよび読出しが行われる。線形アレイの端部に到達したときには、開始点に循環して継続するように、そのステップ動作は循環する（ラップアラウンドする）。言い換えると、バッファ１９０は循環バッファとして動作する。

【００３１】コマンドキュー状態レジスタ３１０がこのプロセスを制御する。レジスタベース（Ｂａｓｅ）３１４およびサイズ（Ｓｉｚｅ）３１１は、関連するキューの場所および長さを記述する。Ｂａｓｅの最下位ビットの設定値により、そのキューは、外部メモリ１２、あるいはレンダリングメモリ１６０のいずれかに存在する。詳細には、最下位ビットが０である場合には、そのキューはＰＣＩアドレス空間内にあり、最下位ビットが１である場合には、そのキューはレンダリングメモリ１６０内にある。

【００３２】サイズレジスタ３１１は、そのキュー内の実際のエントリの数より１だけ小さい値、すなわち２の２乗の値を有する。その後、複雑なラップアラウンドアドレス演算を必要としないように、サイズレジスタをキューポインタを有するマスクとして用いることができる。

【００３３】プロデューサ（Ｐｒｏｄｕｃｅｒ）レジスタ３１６およびコンシューマ（Ｃｏｎｓｕｍｅｒ）レジスタ３１５は、当分野において知られている標準的なリングバッファモデルにしたがって、コマンドキューに指標付けされる。詳細には、プロデューサ３１６レジスタは、新しいコマンドあるいはデータ要素を書き込むことができるキュー内の最初の使用していないエントリを指示する。コンシューマ３１５レジスタは、その時点で処理されている要素を指示する。プロデューサレジスタおよびコンシューマレジスタはいずれもサイズレジスタによってマスクされ、それにより、下位ビットがキューを表すメモリアレイへの相対的な指標を表し、上位ビットが破棄されるようにする。このマスク処理のため、インクリメントのために簡単な算術を用いることができる。ラップアラウンドのための検査を行う必要はない。

【００３４】あるコマンドのための処理が完了するとき、そのコマンドキュー内の次のエントリを指示するように、ポインタロジック３５０がコンシューマ３１５をインクリメントする。そのマスクされた値がプロデューサのマスクされた値に等しい場合には、そのキューは空であり、新しいコマンドがそのキューに書き込まれるまで、構文解析が一時的に中断される。そうでない場合には、次のコマンドｃが読み出され、その処理が開始される。

【００３５】以前に書き込まれたコマンドが処理されている間に、ホストプロセッサ上で実行されるアプリケーションソフトウエアは、コマンドおよびデータをキューに書き込むことができる。ソフトウエアは、プロデューサ３１６のマスクされた値によって示されるエントリにおいて開始するキューに、コマンドおよびデータを書き込むことによりそれを実行する。その書込みが完了するとき、そのソフトウエアは、キューに書き込まれた要素の数だけプロデューサを自動的に更新し、プロデューサがそのキュー内の次の空のエントリを指示するようにする。

【００３６】コマンドキュー内の最後の要素が処理された後、そのキューの始めに次の要素が書き込まれる。当然、そのソフトウエアは、プロデューサがコンシューマに絶対に「追いつか」ないようにしなければならない。もし追いついたなら、ソフトウエアはパース状態機械３３０が未だに処理していないコマンドを上書きしてしまうであろう。

【００３７】レジスタサブベース（ＳｕｂＢａｓｅ）３１３およびサブコンシューマ（ＳｕｂＣｏｎｓｕｍｅｒ）３１２は、以下に記載されるコマンドキュー「サブルーチン」ファシリティを利用可能にする。各コマンド

キューに関連するスクラッチ（Ｓｃｒａｔｃｈ）レジスタ３１７は、ソフトウエアを便利にするために設けられる。このレジスタは、一時的な値を保持することができ、以下に記載される「同期（ｓｙｎｃ）」コマンドで、ロード、格納（ストア）、インクリメントおよび検査されることができる。

【００３８】プロデューサは、以下に記載されるようなコマンドキューのリセットに応答する場合を除いて、ソフトウエアによってのみ更新され、コマンドキューロジック３００によっては決して更新されないことに留意されたい。コンシューマおよびサブコンシューマはコマンドキューロジックによってのみ更新され、決してソフトウエアによっては更新されない。さらに、これらのレジスタは自動的に更新される。それゆえ、これらのレジスタは、プロデューサとコンシューマとの間の比較に関する競合条件を心配することなく、任意の時点で読み出すことができる。

【００３９】コマンドキューにおいて用いるための他のレジスタ図８は、キューを管理し、レンダリングおよびＤＭＡイベントを同期させ、外部メモリを管理する際にソフトウエアによって用いられるための１組のスクラッチおよび一時的レジスタ８００のリストである。レジスタ８００は、スクラッチ８０１、スクラッチダブル８０２およびメモリ管理８０３を含む。これらの各レジスタは、以下に記載される、コマンドロードレジスタ（ｌｏａｄＲｅｇ）およびインクリメントレジスタ（ｉｎｃｒＲｅｇ）を用いて、ソフトウエアあるいはハードウエアによってロード、格納（ストア）あるいはインクリメントされることができる。さらに、そのレジスタは同期（ｓｙｎｃ）コマンドによって検査することもできる。

【００４０】これらのレジスタは、ホストプロセッサに接続されるグラフィックスボードによって生成される埋め込まれた多角形を含むピクセルおよびデプスアレイの管理および転送に対応することを特に意図している。それらは、以下に記載される、コマンドロードレジスタおよびインクリメントレジスタによって自動的にロードおよびインクリメントされる。それゆえ、これらのレジスタとデータを交換し、かつ同期するコマンドキューは、更新と検査との間の競合条件を心配する必要はない。

【００４１】コマンドキューの状態

図４に示されるように、各コマンドキューは、コマンドを構文解析する間に、５つの状態、すなわち、実行状態（ｒｕｎｎｉｎｇ）４０１、コマンド待ち状態（ｗａｉｔｉｎｇ　ｆｏｒ　ｃｏｍｍａｎｄｓ）４０２、同期待ち状態（ｗａｉｔｉｎｇ　ｆｏｒ　ｓｙｎｃ）４０３、停止状態（ｈａｌｔｅｄ）４０４あるいはリセット状態（ｒｅｓｅｔ）４０５のうちの１つの状態にあることができる。状態遷移は、その状態間の指示されたエッジ４１０として示される。たとえば、

【数１】

停止‖リセット（ＨＡＬＴ‖ＲＥＳＥＴ）

を付された遷移は、あるサイクル中に遷移が生じることを示しており、その場合には、キューコントロール（以下を参照）の停止（ｈａｌｔ）およびリセット（ｒｅｓｅｔ）ビットの一方あるいは両方が、その特定のコマンドキューに当てはまることを示す。

【００４２】実行状態４０１では、コマンドキューロジック３００が、コンシューマによって指示されるコマンドで開始して、コマンドキューからコマンドおよびデータを読み出し、次に各コマンドを構文解析し、かつ処理する。

【００４３】コマンド待ち状態４０２では、コンシューマのマスクされた値がプロデューサのマスクされた値に追いついたために、コマンドの構文解析が一時的に中断されている。コマンド構文解析、すなわち実行状態４０１は、プロデューサを更新した後に自動的に再開される。その時点で、そのキューから新しいコマンドが読み出され、処理される。データ転送コマンドを処理することは、ＤＭＡあるいはレジスタ転送を生じさせることを意味し、レンダリングコマンドを処理することは、レンダリングエンジン、たとえばパイプライン１４０においてレンダリング動作を生じさせることを意味する。

【００４４】同期待ち状態４０３では、ある検査を満足するために、以下に記載される同期コマンドがあるレジスタの値を待ち続けているので、コマンド構文解析は一時的に中断されている。コマンド構文解析は、同期コマンドの検査が満足されるようにそのレジスタを更新した後に自動的に再開される。

【００４５】停止状態４０４では、プロデューサあるいはコンシューマの値に関係なく、コマンド構文解析が停止される。停止状態は、図５に関して以下に記載される、キューコントロール（Ｑｕｅｕｅ　Ｃｏｎｔｒｏｌ）レジスタ５００の対応する停止ビットに書込みを行う結果として、あるいは以下に記載される、停止あるいは全停止（ｈａｌｔＡｌｌ）ビットセットを有するコマンドを構文解析する結果として生じる。構文解析は、キューコントロールレジスタ５００の停止ビットをクリアすることによってのみ再開することができる。

【００４６】コマンドキューが停止状態４０４にあるとき、コマンドキューの全状態がそのレジスタセット３１０に含まれる。コマンドキューの隠れた状態あるいは内部状態が保持されることはない。そのキューが空でない場合には、コンシューマが、処理されることになる次のコマンドを指示する。停止ビットがクリアされるとき、コマンドが、バッファ１９０からローカルレジスタアレイ３４０内に再フェッチされる。これにより、ソフトウエアが、そのキューが停止している間にコマンドキューバッファ１９０内のキュー要素を変更でき、再開前に、そのコマンドキューロジック３００が内部バッファ３４０にそれらの変更された要素を確実に読み込むことがで

きるようになる。

【００４７】最後に、リセット状態４０５では、コマンドキューは、定義された初期状態にリセットされている。リセット信号を除去することにより、コマンドキューは停止状態にされる。

【００４８】キューコントロールレジスタ

図５に示されるように、キューコントロールレジスタ５００は、対応するコマンドキューの１つの状態を制御する。このレジスタは、各コマンドキューをリセットするためのビット、コマンドキューを停止させるためのフィールド、およびＤＭＡコマンドを処理することができる場所を厳密に制御するためのビットを与える。そのビットは、厳密（ｓｔｒｉｃｔ）ビット５０１と、リセットＤｍａＯｕｔキュー（ｒｅｓｅｔＤｍａＯｕｔＱｕｅｕｅ）ビット５０２と、リセットレンダーキュー（ｒｅｓｅｔＲｅｎｄｅｒＱｕｅｕｅ）ビット５０３と、リセットＤｍａＩｎキュー（ｒｅｓｅｔＤｍａＩｎＱｕｅｕｅ）ビット５０４と、停止ＤｍａＯｕｔ（ｈａｌｔＤｍａＯｕｔ）ビット５０５と、停止レンダー（ｈａｌｔＲｅｎｄｅｒ）ビット５０６と、停止ＤｍａＩｎ（ｈａｌｔＤｍａＩｎ）ビット５０７とを含む。

【００４９】コマンドキューのリセット

３つのリセットビット５０２～５０４の任意のビットに「１」を書き込むことにより、対応するコマンドキューがリセット状態４０５に入ることができる。そのレジスタ３１０は全て０に設定される。コマンドキューをリセットすることにより、そのキュー内のコマンドの構文解析および処理が中断される。

【００５０】コマンドキューの停止

現在のコマンドの処理が完了するとき、３つの停止ビット５０５～５０７の任意のビットに「１」を書き込むことにより、対応するコマンドキューが停止状態４０４に入ることができる。詳細には、対応するコマンドキューが停止状態に入る前に、長い分散－収集リストを伴うＤＭＡ転送のような、その処理に長い時間を要するコマンド、あるいはレジスタの長いリストを伴うロードレジスタ（ｌｏａｄＲｅｇ）コマンドは、完了するまで実行されるであろう。これにより、ソフトウエアは、たとえばデバッグ中にそのキューを１つずつステップできるようになる。

【００５１】そのキューが、その停止ビットがセットされる時点で待ち（ｗａｉｔｉｎｇ）状態にあるとき、そのキューは直ちに停止状態に入る。その待ち状態が、満足されていない同期コマンドに関する検査の結果であった場合には、同期コマンドが割り込まれ、停止ビットがクリアされるときに同期が再度実行されるように、コンシューマポインタが設定される。

【００５２】停止ビットをクリアすることにより、そのレジスタの値によって、コマンドキューは実行あるいは待ち状態のいずれかに戻ることができる。詳細には、サブベースが０でない場合には、サブコンシューマによって指示されるコマンドは、再フェッチされ、実行される。サブベースが０で、コンシューマがプロデューサと同じでない場合には、コンシューマによって指示されるコマンドは再フェッチされ、実行される。コンシューマおよびプロデューサがコマンドキュー内の同じ要素を指示する場合には、そのキューは空であると見なされ、待ち状態に入る。

【００５３】コンシューマあるいはサブコンシューマによって指示されるコマンドが同期コマンドである場合には、そのコマンドは任意の他のコマンドと同様に再フェッチされ、その検査が再度評価される。その検査が満足されない場合には、そのキューは再び待ち状態に入る。

【００５４】また、コマンドキューは、そのキュー内に含まれるコマンドによって停止される場合もある。要約すると、コマンドキューは４つのうちの任意の方法で停止されることができる。

【００５５】第一に、アプリケーションソフトウエアが、キューコントロールレジスタ５００内の停止ビットを１に設定することができる。そのコマンドキューはその時点のコマンドの終了時に停止することになるが、新しいコマンドを開始しないであろう。その時点のコマンドが同期であり、検査が満足されていない場合には、同期コマンドは中止されるであろう。以下に記載される、キューステータス内の停止ビットは、そのキューが実際に停止状態に入る時点を指示する。

【００５６】第二に、コマンドキューは、自らのコマンドの１つに停止ビットを設定させる場合がある。この場合には、コマンドの処理は完了し、その停止ビットはキューコントロールの対応するエントリに書き込まれ、そのコマンドキューは直ちに停止状態に入る。そのコマンドが同期コマンドである場合には、同期検査が満足されるまで、その停止ビットは書き込まれない。これは、デバッグ中にコマンドを通して１つずつステップするのに有用である。

【００５７】第三に、いくつかのコマンドキューは、そのコマンドの１つにおいて全停止ビットを設定させる場合がある。その場合には、全停止ビットを含むコマンドが完了するとき、キューコントロール（ＱｕｅｕｅＣｏｎｔｒｏｌ）の全ての３つの停止ビットに１が書き込まれる。その１つのコマンドキューは、全停止の代わりにそのコマンドにおいて停止ビットを設定されたかのように、直ちに停止状態に入るであろう。他の２つのキューは、ソフトウエアがその停止ビットに１を書き込んだかのような動きをする。すなわち、そのキューが停止状態に入る前に、その現在のコマンドの処理は完了するが、満足されていない検査を待つ同期コマンドは中断されるであろう。以下に記載されるキューステータス（ＱｕｅｕｅＳｔａｔｕｓ）６００の停止ビットは、各コマンドキューが実際に停止状態に入った時点を示す。

【００５８】第四に、ソフトウエアは、キューコントロールの対応するリセットビットを設定することができる。この場合には、コマンドキューは、リセットビットがクリアされた後に停止する。

【００５９】ＤＭＡコマンドの厳密（Ｓｔｒｉｃｔ）および曖昧（Ｌａｘ）コントロールキューコントロール（ＱｕｅｕｅＣｏｎｔｒｏｌ）レジスタ５００の厳密（ｓｔｒｉｃｔ）ビット５０１は、どのコマンドキューがどのＤＭＡコマンドを処理することができるかを判定する。

【００６０】厳密ビットが１に設定されるとき、ｄｍａＩｎコマンドはＤｍａＩｎキューからのみ処理される場合があり、ｄｍａＯｕｔコマンドは、ＤｍａＯｕｔキューからのみ処理される場合がある。この規則に違反があれば、全ての３つのキューの停止ビットが直ちに１に設定され、中断が発生するようになる。違反したコマンドは処理されず、バイオレーションを含むキューのコンシューマあるいはサブコンシューマは、そのコマンドを指示したままにされる。これは、３つのキューの典型的な使用法を強制し、その使用法に違反するときの誤りを検出する。

【００６１】厳密ビットが０に設定されるとき、あらゆるキューにおいて、あらゆるコマンドが構文解析され、処理されることができる。これは、たとえば、ソフトウエア開発中に、重複した処理が存在しないように、レンダリングエンジンが１つのみのコマンドキューで制御されるときに有用である。また、上記のようなｄｍａＩｎコマンド、レンダーコマンドおよびｄｍａＯｕｔコマンドの典型的な配分以外の態様で、ソフトウエアが３つのキューの中でコマンドを割り振るときに有用である。

【００６２】コマンドキューステータス

図６はキューステータス（ＱｕｅｕｅＳｔａｔｕｓ）レジスタ６００を示す。レジスタ６００は、ステータスロジック３６０によって保持される。このレジスタは、コマンドキューのステータス、およびそれぞれが停止されているか否かを示す。３つの各停止ビット６０４～６０６は、対応するコマンドキューが実際に停止状態４０４に入っていることを示す。

【００６３】各コマンドキューのステータスは、対応する２ビットステータス（ｓｔａｔｕｓ）フィールド６０１～６０３において連続して更新される。そのキューが停止されていないとき、これらは連続して更新する。そのキューが停止するとき、このステータスはラッチされ、停止された時点で、どのコマンドキューが動作中であるかを示す。このステータスは、たとえば、デバッグ中にソフトウエアによって検査されることができる。

【００６４】キューが停止した後、実行状態（ｅｘｅｃｕｔｉｎｇ）のステータスは、停止ビットが設定された時点で、コマンドの処理が進行中であることを意味する。「プロデューサ＝コンシューマのための待ち状態（ｗａｉｔｉｎｇ　ｂｅｃａｕｓｅ　Ｐｒｏｄｕｃｅｒ＝Ｃｏｎｓｕｍｅｒ）」のステータスは、そのキューが空であるため、停止ビットが設定されたときにコマンドが処理されていなかったことを意味する。「同期検査が満足されるのを待つ状態（ｗａｉｔｉｎｇ　ｆｏｒｓｙｎｃ　ｔｅｓｔ　ｔｏ　ｂｅ　ｓａｔｉｓｆｉｅｄ）」のステータスは、満足されていない同期コマンドが、停止ビットを設定することによって中断されたことを意味する。

【００６５】コマンドキューコマンド

図７はキューコマンドのためのフォーマット７００を示す。コマンドは、コマンドヘッダｃ（演算子）と、その後に続くことができるデータ（オペランド）とを含む。コマンドの処理中の典型的な動作は、レジスタをロードあるいはストアし、同期のためにレジスタ値を検査し、ＤＭＡ転送を開始することである。コマンドキューの中からレンダリング動作を開始するために、１つあるいは複数のロードレジスタ（ＬｏａｄＲｅｇ）コマンドによってレンダリングレジスタがロードされる。

【００６６】各コマンドヘッダは以下のフィールドを含む、ｃｍｄフィールド７０１の演算子コードは処理されるコマンドの種類を示す。演算子が、各演算子コードのためのオペランド７０５とともに、以下のサブセクションにおいて列挙される。

【００６７】中断（ｉｎｔｅｒｒｕｐｔ）フィールド７０２が設定されるとき、このコマンドを処理した完了時に中断が通知される。

【００６８】全停止（ｈａｌｔＡｌｌ）フィールド７０３が設定されるとき、キューコントロールレジスタ５００内の３つ全ての停止ビットが、現在のコマンドの完了時に１に設定される。これにより、現在のコマンドキューは停止状態に入り、そのコンシューマあるいはサブコンシューマレジスタが、次に構文解析され、かつ処理されるコマンドを指示できるようになる。他の２つのコマンドキューは、現在のコマンドの処理が完了した後に、停止状態に入る。

【００６９】停止（ｈａｌｔ）フィールド７０４が設定されるとき、現在のコマンドキューは停止状態に入り、コマンド処理が完了した直後に、キューコントロールレジスタの停止ビットが１に設定される。そのコンシューマあるいはサブコンシューマレジスタは、そのキュー内の次のコマンドを指示する。

【００７０】無演算コマンド

無演算（ｎｏｏｐ）コマンドは何も実行せず、完了するのに１クロックサイクルを要する。無演算コマンド中に、コマンドヘッダ内の割込み、停止および全停止ビットが、任意の他のコマンドと同様に構文解析される。このコマンドは、アプリケーションソフトウエアによってプレースホルダとして用いられることができるか、あるいはコマンドの２つの個別のシーケンス間で停止を強制

するために用いることができる。オペランド７０５は無視される。

【００７１】ＤＭＡコマンド２つのＤＭＡコマンド、すなわちｄｍａＩｎおよびｄｍａＯｕｔは、バスロジック２００の中からＤＭＡ転送を生じさせるために与えられる。各ＤＭＡコマンドは、コマンドヘッダ内にカウント（ｃｏｕｎｔ）オペランド７０５を有し、それにそのキュー内のデータ要素の所定の数のリストが後続する。各データ要素は、ホストアドレス（ｈｏｓｔＡｄｄｒｅｓｓ）、ホストアドレスインクリメント（ｈｏｓｔＡｄｄｒｅｓｓＩｎｃｒ）と、レングス（ｌｅｎｇｔｈ）とを含む。どのＤＭＡコマンドが呼び出されるかに関係なく、ホストアドレスおよびレングスによって表されるＤＭＡ転送が完了まで進められる。

【００７２】ＤＭＡコマンドの処理は、キューコントロールレジスタへの停止ビットの書込みによって中断することはできない。そのコマンドの処理は、キューが停止状態に入る前に完了する。しかしながら、ＤＭＡコマンドの処理は、そのコマンドキューがリセットされるときに中断される。この中断は、カウント（ｃｏｕｎｔ）が満了したか否かのいずれの場合でも、レングスバイトの１回のＤＭＡ転送の正常な終了時に生じる。

【００７３】キューコントロールレジスタ５００において厳密ビット５０１が設定される場合には、ｄｍａＩｎコマンドは、ＤｍａＩｎキューからのみ呼び出され、ｄｍａＯｕｔコマンドは、ＤｍａＯｕｔキューからのみ呼び出される場合がある。

【００７４】同期コマンド

本発明の利点として、同期（ｓｙｎｃ）コマンドは、多数のコマンドキューを互いに、かつアプリケーションソフトウエアと同期させるための柔軟な機構を提供する。特定のコマンドキューのための同期コマンドを実行することにより、オペランド７０５によって規定されるレジスタが、検査値（ｔｅｓｔＶａｌｕｅ）オペランド内の値と比較されるようになる（等しい、大きい、小さい等）。比較関係は、検査（ｔｅｓｔ）オペランドによって規定される。同期コマンド自体は、そのコマンドヘッダのみを含む。そのヘッダにはデータ要素は後続しない。

【００７５】比較結果が正しい場合には、そのコマンドキュー内の要素の構文解析が、次のコマンドで続けられる。比較結果に誤りがある場合には、そのコマンドキューは、レジスタ（ｒｅｇ）によって規定されるレジスタが更新され、その比較が正しい結果になるまで、待ち状態になる。すなわち、ある後の時点で、ソフトウエアあるいはいくつかの他のコマンドキューがレジスタを更新するとき、最終的に同期コマンドの検査が満足される。その後、同期コマンドを含むコマンドキューの状態は、待ち状態から実行状態に変化し、次のコマンドが、構文解析および処理のためにフェッチされる。

【００７６】同期コマンドは停止に関して専用である。コマンドキューが未処理の満足されない同期コマンド、すなわち試みられたが、その検査が未だ満足されてなく、かつそのコマンドキューのためのキューコントロール内の停止ビットが、ソフトウエアあるいはいくつかの他のコマンドキューによって１に設定される同期コマンドを有する場合には、その同期コマンドは直ちに中止される。そのコマンドキューのコンシューマあるいはサブコンシューマレジスタは、それが処理されなかったかのように、同期コマンドを指示するために設定される。その後、キューコントロール内の停止ビットがクリアされるとき、同期コマンドはコマンドバッファ１９０からレジスタアレイ３４０に再フェッチされ、再試行される。これは、アプリケーションソフトウエアに、停止状態中の同期（ｓｙｎｃ）に及ぶいくつかの予測可能な制御を与える。ソフトウエアの観点から、同期は進行するか、あるいはそれが未だ試行されていなかった場合には、それが存在していた状態に留まるかのいずれかである。比較値のような隠れた状態は、停止中のコマンドキューには保持されない。

【００７７】また同期コマンドは、同様に動作できるようになる再フェッチビットも含む。詳細には、再フェッチビットが１に設定される場合には、レジスタアレイ３４０内のコマンドキューバッファデータは、それがその時点で停止されたかのように、フラッシュされ、再フェッチされるであろう。これにより、ハードウエアが同期検査が満足されるのを待ち続けている間に、ソフトウエアは、そのキュー内のコマンドを変更できるようになり、変更されたコマンドがレジスタアレイ３４０内に読み込まれることを保証する。これは、たとえば、デバッグのために、および規定された同期条件が満足された後に、正確な値でのみ満たされることができる、コマンドキュー内の次の場所に「ダミー」値が書き込まれる状況の場合に有用である。

【００７８】レジスタコマンド

ストアレジスタ（ｓｔｏｒｅＲｅｇ）、ロードレジスタ（ｌｏａｄＲｅｇ）およびインクリメントレジスタ（ｉｎｃｒＲｅｇ）コマンドによって、レンダリングエンジンレジスタを書き込み、読み出し、インクリメントすることができるようになる。そのレジスタのデータは、ＰＣＩアドレス空間に、あるいはＰＣＩアドレス空間から転送することができる。ストアレジスタコマンドは、そのオペランドとしてレジスタアドレスを、さらにそれに続く、キュー内のデータ要素として６４ビットＰＣＩアドレスを規定する。ロードレジスタコマンドによって、そのオペランド内のレジスタアドレスおよびマスクによって、かつマスクにおいて設定される各ビットのためのキュー内のデータ要素によって規定されるように、可変数の連続したレジスタをロードできるようになる。各データ要素は６４ビットレジスタ値を表す。最後に、イン

クリメントコマンドは、そのオペランドにおいてレジスタアドレスおよびインクリメント値を含んでおり、そのコマンドによって、そのインクリメント値が所定のレジスタに加算される。

【００７９】サブルーチンおよびリターン

サブルーチン（Ｓｕｂｒｏｕｔｉｎｅ）およびリターン（Ｒｅｔｕｒｎ）コマンドによって、「インライン」、すなわち「コマンドサブルーチン」ではない他のキューコマンドを処理できるようになる。サブルーチンコマンドによって、コマンドサブルーチンを処理できるようになり、その後、正規の順序でキューからのコマンドを処理し続ける。サブルーチンコマンドヘッダに続くデータ要素は、メモリアドレスとして取り扱われ、そのサブルーチンを呼び出すコマンドキューのサブベースレジスタ３１３に書き込まれる。リターンアドレスはコンシューマレジスタ３１５に保持され、サブルーチンコマンドおよびそのデータ要素に続くメインコマンドキュー内の要素を指示する。サブルーチンコマンドは、コマンドキューバッファの場合のように、コマンドおよびデータを含む要素のリストとして格納される。しかしながら、サブルーチンコマンドキューバッファは循環バッファではない。それは単に、リターンコマンドによって終了される、コマンドの直線的なリストである。

【００８０】好ましい実施形態では、サブルーチンコマンド自体を除く、任意のコマンドがサブルーチンコマンドキュー内に置かれる場合がある。すなわち、サブルーチンは他のサブルーチンを呼び出さない。しかしながら、他の実施形態では、サブルーチンは、付加的なオンチップサブベース（ｓｕｂＢａｓｅ）レジスタを設けることにより、あるいはチップ外のメモリにデータを格納することにより、他のサブルーチンを呼び出すこともできる。

【００８１】サブルーチンの大部分は、典型的にはＤＭＡコマンドのために用いられる。これにより、アプリケーションソフトウエアは、アクセス情報を、ある種類の動作のためのコマンドキューバッファにおいてテンプレートから都合よく切り離しておくことができる。サブルーチンは、そのコマンドキューが停止状態になるときに、中断される場合がある。そのコマンドキューのための停止ビットがクリアされるとき、サブベースレジスタは０ではないことに留意されたい。次のコマンドは、メインコマンドキューからの要素からではなく、サブコンシューマによって示されるサブルーチンコマンドリスト内の要素からフェッチされる。

【００８２】リターンコマンドによって、構文解析は、コンシューマによって指示されるエントリにおいてメインコマンドキューから再開されるようになる。リターンコマンドは、サブコンシューマおよびサブベースをいずれも０にクリアし、その結果、停止状態の後にコマンド処理を再開する場所に関して混乱は生じない。サブルーチンコマンドの停止ビットが設定される場合には、そのサブルーチンをコールした後、そのサブルーチンの最初のコマンドを実行する前に、コマンドキューが停止する。停止されたとき、サブベースは、サブルーチンコマンドに続くコマンドのアドレスを指示し、サブコンシューマは０であり、コンシューマは、サブルーチンからのリターンの後に再開するためのコマンドを指示する。リターンコマンドにおいて停止ビットが設定される場合には、そのサブルーチンからリターンした後、サブルーチンコールに続く最初のコマンドを用いて進行する前に、コマンドキューは停止する。この場合には、サブベースおよびサブコンシューマはクリアされ、コンシューマが次に処理されることになるコマンドを指示する。

【００８３】

【発明の効果】こうして、本発明は、レンダリング動作から入力および出力データ転送を分離し、データ転送とレンダリング動作との間を同期させることができるコマンドキュー構造を実現する。これは、転送と動作とに長時間を要する、すなわち多くのクロックサイクルを要するときに有利である。これらの動作を別々に取り扱うことにより、特に正確な動作のために必要とされる場合、たとえばレンダリングが完了するまで、システムメモリに画像をコピーするのを遅らせる場合を除いて、長時間を要する動作が、別の各動作を待つ必要がなくなる。これらの同期遅延は、ソフトウエアの介入を必要とせず、コマンドキューコントロールロジック内で完全に処理することができる。

【００８４】本明細書に記載されるコマンドキュー構造は、実際に、任意の種類のグラフィックレンダリングエンジン、あるいは任意の他のハードウエアアクセラレータに適用することができる。本発明は、ホスト処理、サブシステム処理、および広帯域ＤＭＡ転送を同時に必要とする多数のレンダリング動作が存在する際に特に有利である。

【００８５】本発明は好ましい実施形態を例示しながら記載されてきたが、本発明の精神および範囲から逸脱することなく、種々の他の適用形態および変更形態を実施できることは理解されたい。それゆえ、添付の請求の範囲の目的は、全てのそのような変形形態および変更形態を、本発明の真の精神および範囲内に入るように網羅することである。

【図面の簡単な説明】

【図１】　本発明によるコマンドキューを用いるレンダリングエンジンのブロック図である。

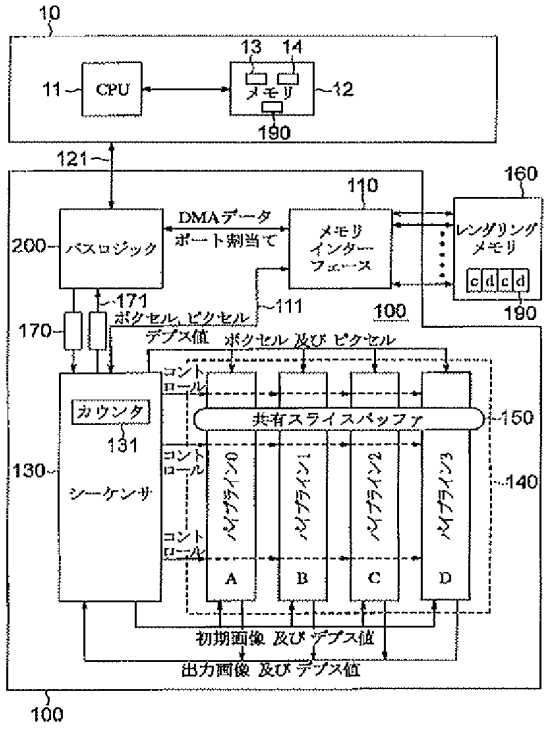
【図２】　コマンドキューロジックを備えるバスロジックのブロック図である。

【図３】　本発明によるコマンドキューを実装する好ましい実施形態のブロック図である。

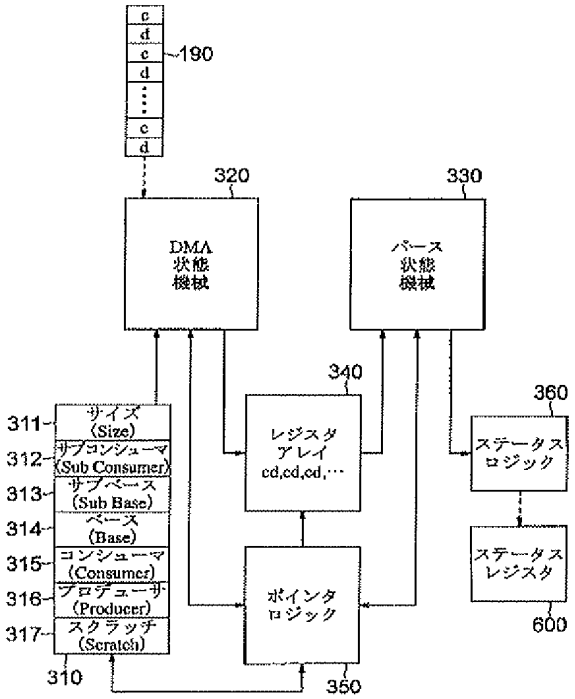
【図４】　コマンドキュー状態および状態遷移のグラフを示した説明図である。

【図５】　キューコントロールレジスタのテーブルを示した説明図である。
【図６】　キューステータスレジスタのテーブルを示した説明図である。
【図７】　キューコマンドヘッダのテーブルを示した説明図である。
【図８】　同期のために用いられるコマンドキューレジスタのテーブルを示した説明図である。
【符号の説明】
１０　ホストコンピュータ、１１　ＣＰＵ、１２　ホストメモリ、１３　グラフィカルアプリケーションソフトウエア、１４　グラフィックドライバ、１００レンダリングエンジン、１１０　メモリインターフェース、１３０　シーケンサ、１４０　並列パイプライン、１５０　スライスバッファ、１６０　レンダリングメモリ、１７０　ＦＩＦＯ、１９０　コマンドキューリングバッファ、２００　バスロジック、２０１　ＤｍａＩｎコマンドキューロジック、２０２　レンダリングコメンドキューロジック、２０３　ＤｍａＯｕｔコマンドキューロジック、２１０　ＤＭＡインターフェース、２２０　ターゲットインターフェース、２３０　レンダリングレジスタ、２４０　仲裁ロジック、３００　ロジック、３１０　コマンドキュー状態レジスタ、３２０　ＤＭＡ状態機械、３３０　パース状態機械、３４０　レジスタアレイ、３５０　ポインタロジック、３６０　ステータスロジック、４０１　実行状態、４０２　コマンド待ち状態、４０３　同期待ち状態、４０４　停止状態、４０５　リセット状態、５００　キューコントロールレジスタ、６００　ステータスレジスタ、７００　フォーマット、８００コマンドキューレジスタ。

【図１】



【図３】

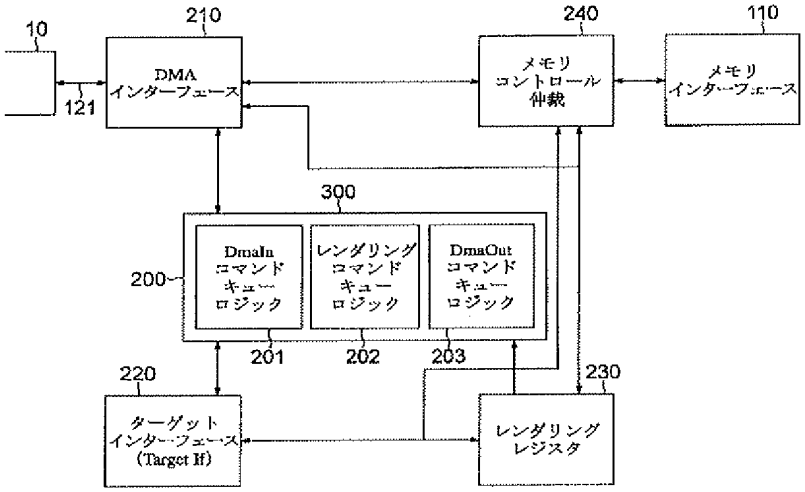


【図８】

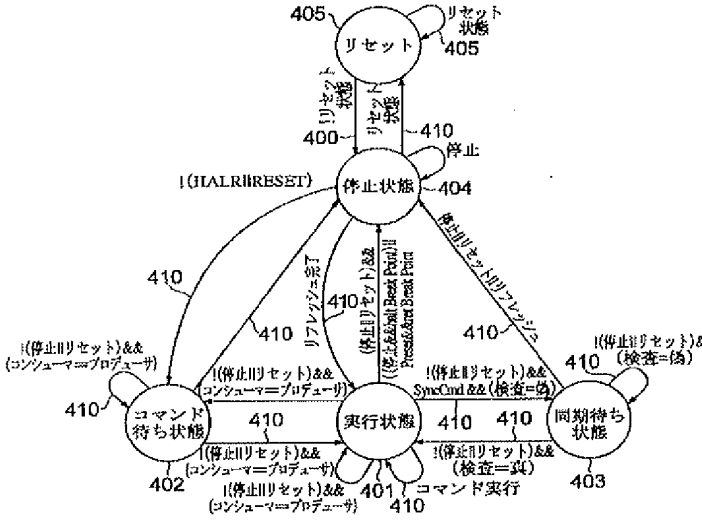
	レジスタ	タイプ	説明
801	スクラッチ	R/W	ソフトウエア使用のためのレジスタ
802	スクラッチダブル	R/W	ソフトウエア使用のための長いレジスタ
803	メモリ管理	R/W	外部のPCIメモリのどこかに存在するバッファを管理するための専用の同期レジスタ

800

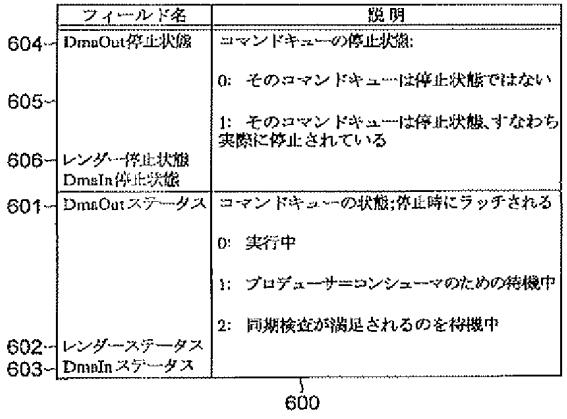
【図２】



【図４】


405
リセット
リセット
状態
405
リセット状態
リセット状態
400
410
停止
!(HALR||RESET)
停止状態
404
410
410
410

【図６】



	フィールド名	説明
604	DmaOut停止状態	コマンドキューの停止状態: 0: そのコマンドキューは停止状態ではない 1: そのコマンドキューは停止状態、すなわち実際に停止されている
605		
606	レンダー停止状態 DmaIn停止状態	
601	DmaOutステータス	コマンドキューの状態;停止時にラッチされる 0: 実行中 1: プロデューサ=コンシューマのための待機中 2: 同期検査が満足されるのを待機中
602	レンダーステータス	
603	DmaInステータス	

600

【図７】

	フィールド名	説明
701	コマンド(Cmd)	実行されるコマンド 8: サブルーチンコールからのリターン 7: サブルーチン 6: インクリメントレジスタ 5: ストアレジスタ 4: ロードレジスタ 3: 同期 2: dmaIn 1: dmaOut 0: 無演算
702	中断(interrupt)	このコマンドが完了したとき、中断を通知する
703	全停止(haltAll)	このコマンドが完了したとき、3つ全てのキューを停止する
704	停止(halt)	このコマンドが完了したとき、このキューを停止する
705	オペランド(operands)	特定の解釈がコマンド(cmd)に依存する

700

フロントページの続き

(72)発明者 ケニス・ダブリュ・コレル
アメリカ合衆国、マサチューセッツ州、ランカスター、ルネンバーグ・ロード 2193
(72)発明者 スティーヴン・カーズナー
アメリカ合衆国、マサチューセッツ州、メドフォード、アーリントン・ストリート 21、アパートメント ナンバービー
(72)発明者 ヒュー・シー・ラウアー
アメリカ合衆国、マサチューセッツ州、コンコード、ボーダー・ロード 69
(72)発明者 ラリー・ディー・シーラー
アメリカ合衆国、マサチューセッツ州、ボイルストン、リンデン・ストリート 198

Fターム(参考) 5B013 DD01
5B057 CH02 CH05 CH06 CH11 CH14 CH16
5B080 CA03 CA04 CA05 CA07 CA08